# PARALLEL PAGE RANK

**MAYA THAPA**
**(TEQIP_INTERN)**

**About:**
**Education:**

MAYA THAPA is an undergraduate in Computer Science and Engineering at Atal Bihari Vajpayee Government Institute of Engineering and Technology, Shimla, Himachal Pradesh.

## Working under faculty supervisor:

Dr. Sathya Peri
Associate Professor
Department of Computer Science and Engineering, IIT Hyderabad

**Research Interests**

Parallel Computing : Software Transactional Memory, Concurrent Data-Structures
Distributed Systems: Blockchain Design, Mining in Bitcoin & Ethereum Networks, Peer-To-Peer Computing, Grid Computing
*. Algorithm Analysis
*. Networking Algorithms.

## Internship provided by:

Technical Education Quality Improvement Programme (TEQIP) Cell, IIT Hyderabad .

## Internship Duration:

One month (11th June ’19 – 10th July ’19)

## Abstract:

The internet can be used to access a huge 'library' of information from millions of websites around the world that make up the World Wide Web. Around 90% of the world’s population uses search engines for getting

relevant information. Thus, in this huge collection of websites the correct data retrieval at the least time domain is the most important task.
For this, Pagerank is used which is a method for rating Web pages according to their relevance and importance.
Since the internet has large distributions of weblinks, collaboration of partial results after processing is a major issue. So here the approach aims at providing a parallel solution to it. Hence, designing a PageRanking algorithm efficiently modified for parallel environment that achieves higher accuracy and consumes less time to evaluate the PageRank of a given webgraph.


## Acknowledgement:

I am very thankful to the TEQIP team and also to the MHRD, Govt. of India for such an excellent internship program.

I would like to thank my guide Dr. Sathya Peri at IIT Hyderabad.


## Introduction:

The World Wide Web created many new challenges for information retrieval. It is very large and heterogeneous. In order to tackle with relative importance of web pages Google Co-founder – Larry Page along with Serge Bin developed a method known as “page rank”.
Page rank deals with link structure of web pages.It ranks the pages according to the number of back links pointing to them. Here, back link refer to the number of links pointing to the specific web pages or can be referred to as incoming links or hyperlinks. For this purpose Pagerank algorithm is established which scans all possible webpages and then calculates the rank accordingly given by formula.
Traditional approaches of ranking algorithm proposed - link and content oriented which do not contain the user usage trend. But pagerank takes into account the number of visits to inbound links of web pages which is useful to display most valuable pages on the top of the result on the basis of user behavior.

There are many challenges one encounters while calculating PageRank value of web pages. The first difficulty is that the input data set is quite huge; therefore it requires a lot of managing and computing effort.
The second problem comes from the property of the Web: it is dynamic. This
property is reflected when the content of web pages are to be changed along with the time. This leads to the change of hyperlinks in the pages and therefore the change of the Web's structure. Moreover, the size of the World Wide Web, which is determined by the number of web page, increased rapidly with billions of web pages being created every year. To make PageRank values always up-to-date despite these changes, PageRank calculation should be carried out in as short a time period as possible.

## Background:

## What is page rank ? And how it is calculated?

Pagerank is method whose motive is to determine how popular the page is based on the other pages link to it. Page rank evaluates the quality and quantity of links to a web page to determine relative score of that page's importance and authority on a 0 to 10 scale. If a web page has a lot of links from large websites that also ranks well, then the original web page is given a high ranking.
PageRank is computed iteratively; while at the very beginning (time t) all n pages have the same score 1/n, as time passes (time t + 1) the i-th page has instead the following PageRank value:

$$p_{i,t+1} = \frac{1-d}{n} + d \sum_{j=1}^{n} A_{ij} P_{j,t}$$

In this formula d represents a damping factor usually set to a value between 0 and 1 (the implementation uses d = 0:85); Aij represents the (i, j)-th element of the state transition probability matrix, an incidence matrix with a stochastic structure:

A = A11…… …….A1n
A21… ….. … …..
… … … …. … … ..
An1… … … Ann

More specifically the element Aij represents the transition probability that the user in page i will navigate to page j, and is computed as:

$A_{ij}$ =

| | |
|---|---|
| $1/O_j$ | if (j; i) 2 E |
| 1/n | if (j; i) =2 E and $O_j = 0$ |
| 0 | otherwise |

Notice that Oj is the number of links called outlinks from page j to other pages,assuming the user clicks links in the i-th page randomly and uniformly without using the back button of the browser or typing an URL; pages with no outlinks are indeed possible, and are called dangling pages. Since a page is typically linked to a limited set of other pages the matrix A, albeit very large, is usually sparse: for this reason we can store it using the so-called Compressed Sparse Row representation.

- **Damping factor**

The PageRank theory holds that an imaginary surfer who is randomly clicking on the links will eventually stop clicking.
The probability, at any step, that the person will continue is a damping factor *d*. Various studies have tested different damping factors, but it is generally assumed that the damping factor will be set around 0.85.
The damping factor is subtracted from 1 (and in some variations of the algorithm, the result is divided by the number of documents (*N*) in the collection) and this term is then added to the product of the damping factor and the sum of the incoming PageRank scores.
So any page's PageRank is derived in large part from the PageRanks of other pages.

## My Contribution:

After detailed study of the PageRanking algorithms, we have concluded that the PageRank formula, cannot be altered anymore. Hence to achieve an improved parallel algorithm we have used OpenMP API.

## Introduction to Openmp

- **OpenMP** (**Open Multi-Processing**) is an application programming interface (API) that supports multi-platform shared memory multiprocessing programming in C, C++, and Fortran on most platforms, instruction set architectures and operating systems, including Solaris, AIX, HP-UX, Linux, macOS, and Windows
- Scheduling clauses:
- This is useful if the work sharing construct is a do-loop or for-loop. The iteration(s) in the work sharing construct are assigned to threads according to the scheduling method defined by this clause. The three types of scheduling are:
- *static*: Here, all the threads are allocated iterations before they execute the loop iterations. The iterations are divided among threads equally by default. However, specifying an integer for the parameter *chunk* will allocate chunk number of contiguous iterations to a particular thread.
- *dynamic*: Here, some of the iterations are allocated to a smaller number of threads. Once a particular thread finishes its allocated iteration, it returns to get another one from the iterations that are left. The parameter *chunk* defines the number of contiguous iterations that are allocated to a thread at a time.
- IF control*if*: This will cause the threads to parallelize the task only if a condition is met. Otherwise the code block executes serially.

## Pseudo code of page rank algorithm

/* PSEUDO CODE FOR IMPLEMENTATION OF PAGE RANK USING CSR (COMPRESSERD SPRASE ROW) */

STEPS:

1.Open the file data set(which containing graph data)

2.if file is empty

```
 {
  print("error")
   }
else
  {
 read the data from file
```

}

3.Define CSR structure

Compressed sparse row format:

§ Value vector: contains 1.0 if an edge exists in a certain row

§ Columnindex vector: contains the column index of the corresponding value in 'value‘.

§ Rowpointer vector: points to the start of each row in 'columnindex'

4.Initialise a vector p

for i to n

{

p[i]=1/n

}

5.#pragma omp parallel for schedule(static/dynamic) if (parallel) num_threads (numthreads)

6.set damping factor d=0.856.Calculate page rank if damping factor is not given

{

p_new[]=p_new[]+value[]*p[]

}

else

{

p_new=(1-d/n)+d*sumation of (PR(y)/outlink(y))

y->x

where x =page rank of x is to be calulated , y=number of links to x

}

7.Terminate the program if consecutive instances of pagerank vector are almost identical

8.calculate iteration

9.print the page rank

10.track time for execution

## Result:

After implementing both sequential and parallel page rank algorithm on a given data set we get the following result:

Page rank comparison:

| Pages(nodes) | Sequential pagerank implementation | Parallel pagerank implementation |
|---|---|---|
| 1 | 0.006624 | 0.004498 |
| 2 | 0.005401 | 0.004498 |
| 3 | 0.004820 | 0.002586 |
| 4 | 0.004610 | 0.002568 |

Time comparison:

| | Sequential | Parallel |
|---|---|---|
| Time | 0.00425 | 0.00339 |

Iteration comparison:

| | Sequential | Parallel |
|---|---|---|
| Iteration | 19 | 13 |

As we can see that parallel page rank implementation leads to much faster execution of the program and also the number of iterations required for reaching at a stable page rank is also less.
Hence, parallel implementation leads to an optimal utilization of resources as well as this increases the performance in terms of time taken to compute pagerank of a given dataset.

# Conclusion:

- In this paper, we have taken some pages and calculate page rank. PageRank is a global ranking of all web pages, regardless of their content.
- Using PageRank, we are able to order search results so that more important and central Webpages are given preference. In experiments, this turns out to provide higher quality search results to users. The intuition

behind PageRank is that it uses information which is external to the Web pages themselves - their backlinks, which provide a kind of peer review. Furthermore, back links from “important" pages are more significant than back links from average pages. This is encompassed in the recursive definition of PageRank .

- PageRank could be used to separate out a small set of commonly used documents which can answer most queries. The full database only needs to be consulted when the small database is not adequate to answer a query. Finally, PageRank may be a good way to help and representative pages to display for a cluster center.
- We have found a number of applications for PageRank in addition to search which include trace estimation, and user navigation. Also, we can generate personalized PageRanks which can create a view of Web from a particular perspective.
- Overall, our experiments with PageRank suggest that the structure of the Web graph is very useful for a variety of information retrieval tasks.

## Learning Accomplishments:

I started my internship with not having any background from web page ranking and threads and from that day to now I have learned a lot. For developing purposes, I have learned to implement different data structures like stacks , queues, BFS and DFS. After this I have done some hands-on experience  on pthreads .And finally able to implement sequential as well as parallel page rank.